Visual Basic 6.0 ▸▸▸ MICROSOFT

PC WORLD PERÚ

El lenguaje de programación Visual Basic proporciona todas las herramientas necesarias para el desarrollo rápido de aplicaciones. Se le podría definir como el método que se utiliza para desarrollar la interfase gráfica de usuario y crear aplicaciones en la red.

▸ **1. La barra de títulos:** La barra de menús y la barra de herramientas de Visual Basic 6.0 en modo Diseño (parte superior de la pantalla).

▸ **2. Ventana de Propiedades:** Muestra las propiedades del objeto seleccionado, o del propio formulario (en el centro a la derecha). Si esta ventana no aparece, se puede hacer visible con la tecla <F4>.

▸ **3. Formulario (form):** En gris. Se pueden ir situando los controles (en el centro) y está dotado de una rejilla (grid) para facilitar la alineación de los mismos.

▸ **4. Ventana de proyecto:** Muestra los formularios y otros módulos de programas que forman parte de la aplicación (arriba a la derecha).

▸ **5. Caja de herramientas (toolbox):** Muestra los controles disponibles (a la izquierda de la ventana).

▸ **6. Ventana FormLayout:** Permite determinar la forma en que se abrirá la aplicación cuando comience a ejecutarse. (abajo a la derecha).

▸ Existen otras ventanas para edición de código (Code Editor) y para ver variables en tiempo de ejecución con el depurador o Debugger (ventanas Immediate, Locals y Watch). Todo este conjunto de herramientas y de ventanas es lo que se llama un entorno integrado de desarrollo o IDE (Integrated Development Environment).

Construir aplicaciones con Visual Basic 6.0 es muy sencillo: basta crear los controles en el formulario con ayuda de la Caja de herramientas (toolbox) y del mouse, establecer sus propiedades con ayuda de la ventana de propiedades y programar el código que realice las acciones adecuadas en respuesta a los eventos o acciones que realice el usuario.

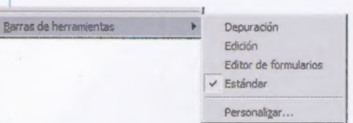
▸▸▸ REQUISITOS DE SISTEMA

▸ **Para usar Microsoft Visual Basic 6.0 se requiere:** PC con procesador Pentium (recomendado procesador Pentium 90 Mhz o superior); sistema operativo Microsoft Windows 95 o superior, o sistema operativo Microsoft Windows NT versión 4.0 con Service Pack 3 o superior; Microsoft Internet Explorer 4.01 Service Pack 1 (incluido); CD-ROM drive; monitor VGA o de alta resolución (recomendado Super VGA); Microsoft Mouse o dispositivo compatible.

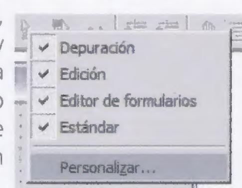
▸ **Para la edición Empresarial se requiere:** 32 MB de RAM para Windows 95 o posterior (48 MB recomendado); 32 MB para Windows NT 4.0 (48 MB recomendado); 116MB de espacio en disco para la instalación típica, 135MB para la instalación completa; espacio de disco duro adicional requerido para los siguientes productos y tecnologías: Internet Explorer: 43/59 MB (típica/completa), MSDN: instalación típica 57/493 MB (típica/completa).

▸ **Para la edición Profesional se requiere:** 24 MB de RAM para Windows 95 o superior (48 MB recomendado); 32 MB para Windows NT 4.0 o superior (48MB recomendado); 76MB de espacio en disco para la instalación típica, 94MB para la instalación completa; espacio adicional espacio de disco duro requerido para los siguientes productos: Internet Explorer: 43MB/59MB (típica/completa), MSDN: 57/493MB (típica/completa).

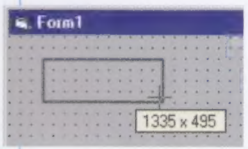
▶▶▶ INICIANDO UN PROYECTO



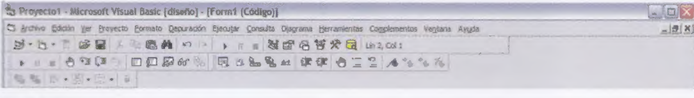
Cuando se ingresa a Visual Basic 6.0, se presenta la barra de menús, similar a la de otra aplicación de Windows. Bajo dicha barra aparecen las barras de herramientas, con una serie de botones que permiten acceder fácilmente a las opciones más importantes de los menús. En Visual Basic 6.0 existen cuatro barras de herramientas: Depuración (Debug), Edición (Edit), Editor de Formularios (Form Editor) y Estándar (Standard). Por defecto solo aparece la barra Estándar. Haciendo clic con el botón derecho sobre una barra aparece un menú contextual que permite ocultarlas o visualizarlas. Las barras pueden ser modificadas, añadiendo o eliminando botones.

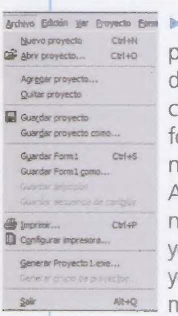


En la barra de herramientas Estándar también se puede ver a la derecha dos recuadros con números, que representan cuatro propiedades del formulario referentes a su posición y tamaño: Top y Left, que indican la posición de la esquina superior izquierda del formulario, y Height y Width, que describen el tamaño del mismo en unidades llamadas twips, que son la vigésima parte de un punto.



Las dimensiones de un control aparecen también cuando con el mouse se arrastra sobre el formulario. Los botones de la barra de herramientas Estándar responden a las funciones más importantes: abrir y/o guardar nuevos proyectos, añadir formularios, hacer visibles las distintas ventanas del entorno de desarrollo, etc. Todos los botones tienen su correspondiente comando en los menús.



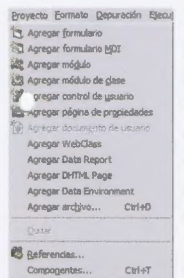


▶ **Menú Archivo:** La novedad más importante es la distinción entre proyectos y todos los demás archivos. Un proyecto reúne y organiza todos los archivos que componen el programa o aplicación (hace la función de una carpeta que contiene apuntes). Estos archivos pueden ser formularios, módulos, clases, recursos, etc. Visual Basic 6.0 permite tener más de un proyecto abierto simultáneamente. Con el comando Agregar Proyecto se añade un nuevo proyecto en la ventana Administrador de Proyecto.Con los comandos Abrir Proyecto o Nuevo Proyecto se abre o se crea un nuevo proyecto, pero cerrando el o los proyectos que estuvieran abiertos previamente. En este menú está el comando *Generar ProjectName.exe*, que permite crear ejecutables.

▶ **Menú Edición:** Este menú permite acceder a funciones como insertar, eliminar, seleccionar columnas o campos de una tabla de una base de datos, establecer claves primarias y eliminar tablas de una base de datos.

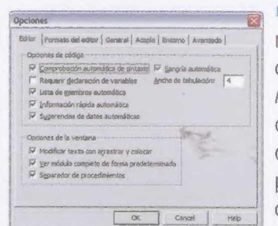
▶ **Menú Ver:** Este menú permite hacer aparecer en pantalla las distintas ventanas del entorno de desarrollo, así como acceder a un formulario o al código relacionado con un control (que también aparece si hace clic dos veces en dicho control), y manejar funciones y procedimientos.

▶ **Menú Formato:** Contiene opciones para controlar el aspecto de la aplicación (alinear controles, espaciarlos uniformemente, etc.).



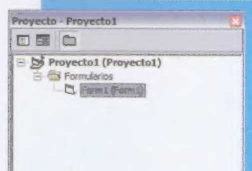
▶ **Menú Proyecto:** Permite añadir distintos tipos de elementos a un proyecto (formularios, módulos, etc.). Con Proyecto/Propiedades se puede elegir el tipo de proyecto y determinar el formulario con el que se arrancará la aplicación Establecer como Inicial (Startup Object). Con el comando Componentes se pueden añadir nuevos controles a la Barra de Herramientas que aparece a la izquierda de la pantalla.

▶ **Menús Depuración y Ejecutar:** Permiten controlar la ejecución de las aplicaciones. Con Depuración se puede ver en detalle cómo funcionan, ejecutando paso a paso, yendo hasta una línea de código determinada, etc. Esto es especialmente útil cuando haya que encontrar algunos errores ejecutando paso a paso, o viendo resultados intermedios.



▶ **Menú Herramientas:** Están los comandos para arrancar el Menu Edición y para establecer las opciones del programa. En *Herramientas•Opciones* están las opciones relativas al proyecto en el que se trabaja. Entorno determina las propiedades del entorno del proyecto, como las opciones para actualizar los archivos. En General se establece lo referente a la retícula o grid. Editor permite declarar las variables y otras opciones de edición. Formato del Editor permite seleccionar el tipo de letra y los códigos de color. Avanzado hace referencia a la opción de utilizar Visual Basic 6.0 en dos formatos SDI (Single Document Interface) y MDI (Multiple Document Interface).

▶▶▶ LA VENTANA DEL PROYECTO



▶ Esta ventana permite acceder a los distintos formularios y módulos que componen el proyecto. Desde ella se puede ver el diseño gráfico de dichos formularios (botón Ver Objeto), y también permite editar el código que contienen (botón Ver Código). Estos botones están situados en la parte superior de la ventana, debajo de la barra de títulos.

Los módulos estándar (archivos *.bas) contienen solo código que, en general, puede ser utilizado por distintos formularios y/o controles del proyecto, e incluso por varios proyectos. Por ejemplo, puede prepararse un módulo estándar de funciones matemáticas que sea de utilidad general. Normalmente contienen siempre algunas declaraciones de variables globales o Public, que serán accesibles directamente desde todos los formularios.

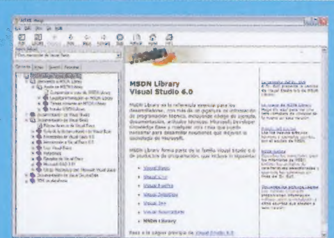
Los módulos de clase (archivos *.cls) contienen clases definidas por el usuario. Las clases son como formularios o controles complejos, sin interfase gráfica de usuario.

▶▶▶ OBTENIENDO AYUDA

▶ Visual Basic 6.0 dispone de un sistema de ayuda (Help) excelente. En esta versión la ayuda se ofrece a través de una interfase de usuario similar a la de Internet Explorer.

Estando seleccionado un control, una propiedad o un formulario, o estando seleccionada una palabra clave en una ventana de código, esta ayuda se puede utilizar de modo sensible al contexto pulsando la tecla <F1>.

También se puede ver toda la información disponible de modo general y ordenada por temas con el comando Help.Contents.

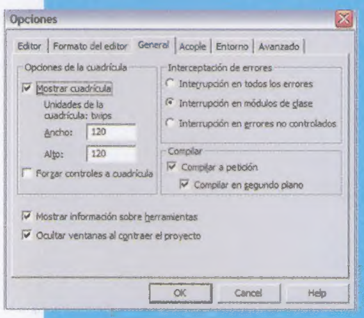


LAS HERRAMIENTAS (TOOLBOX)

La caja de herramientas incluye los controles con los que se puede diseñar la pantalla de la aplicación. Estos controles son por ejemplo botones, etiquetas, cajas de texto, zonas gráficas, etc. Para introducir un control en el formulario simplemente hay que dar clic en el ícono adecuado de la Caja de Herramientas (toolbox), y colocarlo en el formulario con la posición y el tamaño deseado, haciendo clic y arrastrando con el mouse. Haciendo Clic dos veces sobre el ícono de un control, aparece este en el centro del formulario y se puede modificar su tamaño y/o trasladar con el mouse a donde se desee. El número de controles que pueden aparecer en esta ventana varía con la configuración del sistema. Para introducir nuevos componentes se utiliza el comando Componentes en el menú Proyecto, con lo cual se abre el cuadro de diálogo correspondiente.

HERRAMIENTAS	DESCRIPCIÓN	HERRAMIENTAS	DESCRIPCIÓN
	Puntero. Regresa el mouse a su estado normal.		*VScrollBar.* Control para hacer las veces de una barra vertical.
	PictureBox. Control que permite colocar una imagen sobre el formulario		*Timer.* Control para manejos de intervalos de tiempo.
	Label. Control que permite etiquetar o rotular un dato en el formulario.		*DriveListBox.* Control para mostrar las unidades de disco de la PC.
	Textbox. Control para ingreso de datos en un formulario.		*DirListBox.* Control para mostrar los directorios de una unidad de disco.
	Frame. Control que funciona como contenedor de otros controles		*FileListBox:* Control para mostrar archivos de un directorio.
	CommandButton. Control que permite hacer acciones de comando como por ejemplo accionar procedimientos.		*Shape.* Control para insertar figuras rectangulares, circulares, etc.
	CheckBox. Control que permite marcar o desmarcar una opción en un formulario.		*Line.* Permite crear una línea.
	OptionButton. Control que permite seleccionar una sola opción de un grupo de opciones de este mismo control.		*Image.* Control similar al PictureBox, pero a diferencia se puede acomodar al tamaño real de la imagen.
	ComboBox. Control para mostrar una lista de selección a modo de menú de opciones.		*Data.* Control que permite conectarse a una base de datos y obtener la información de tablas o consultas.
	ListBox, Control que muestra una lista de selección y permite escoger uno o mas elementos.		*Ole.* Control que permite importar vía OLE objetos de otras aplicaciones como por ejemplo un Excel o un Word.
	HScrollBar. Control para hacer las veces de una barra horizontal.		

FORMULARIOS (FORMS) Y MÓDULOS



Los formularios son las zonas de la pantalla sobre las que se diseña el programa, y sobre las que se sitúan los controles o herramientas de la caja de Herramientas. Al ejecutar el programa, el form se convertirá en la ventana de la aplicación, donde aparecerán los botones, el texto, los gráficos, etc.

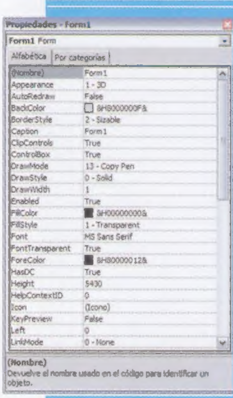
Para lograr una mejor presentación existe una malla o retícula (grid) que permite alinear los controles manualmente de una forma precisa (evitando tener que introducir coordenadas continuamente). Esta malla solo será visible en el proceso de diseño del programa; al ejecutarlo no se verá. De cualquier forma, se puede desactivar la malla o cambiar sus características en el menú *Herramientas•Opciones•General*, cambiando la opción Forzar controles a Cuadricula.

Exteriormente, los formularios tienen una estructura similar a la de cualquier ventana. Sin embargo, también poseen un código de programación que estará escrito en Basic, y que controlará algunos aspectos del formulario, sobre todo en la forma de reaccionar ante las acciones del usuario (eventos). El formulario y los controles en él situados serán el esqueleto o la base del programa. Una aplicación puede tener varios formularios, pero siempre habrá uno con el que arrancará la aplicación; este formulario se determina a partir del menú *Proyecto•Propiedades*, en Objeto Inicial.

Resumiendo, cuando se vaya a crear un programa en Visual Basic 6.0 habrá que dar dos pasos:

- Diseñar y preparar la parte gráfica (formularios, botones, menús, etc.)
- Realizar la programación que gestione la respuesta del programa ante los distintos eventos.

LA VENTANA DE PROPIEDADES (PROPERTIES)

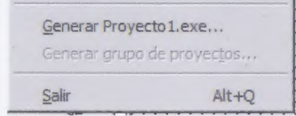


Todos los objetos Visual Basic 6.0 tienen unas propiedades que los definen: su nombre (Name), su etiqueta o título (Caption), el texto que contiene (Text), su tamaño y posición, su color, si está activo o no (Enabled), etc. Todas estas propiedades se almacenan dentro de cada control o formulario en forma de estructura (similar a las del lenguaje C). Por tanto si, por ejemplo, en algún momento se quiere modificar el nombre de un botón, basta con hacerlo en la ventana de propiedades (al diseñar el programa), o en el código en Basic (durante la ejecución), mediante el operador punto (.), en la forma: Boton1.Name = "NuevoNombre"

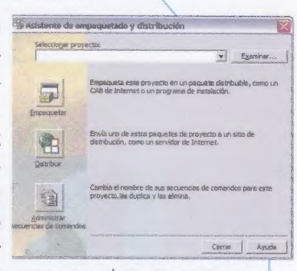
Para realizar una modificación de las propiedades de un objeto durante el diseño del programa, se activará la ventana de propiedades (con el menú, con el botón de la barra de herramientas o pulsando <F4>). Esta ventana tiene dos lengüetas, que permiten ordenar las propiedades alfabéticamente o por categorías. Con ayuda de la barra de desplazamiento se localizará, la propiedad que se quiera modificar, y al hacer clic sobre ella puede activarse un menú desplegable con las distintas opciones, o puede modificarse directamente el valor de la propiedad. Si esta tiene solo unos valores fijos (por ejemplo, los colores), puede abrirse un cuadro de diálogo para elegir un color, o el tamaño y tipo de letra que se desee si se trata de una propiedad Font. La ventana Posición del Formulario permite determinar la posición en la que el formulario aparecerá sobre la pantalla, cuando se haga visible al ejecutar la aplicación.

CREANDO PROGRAMAS EJECUTABLES

Una vez finalizada la programación de la nueva aplicación, la siguiente tarea suele consistir en la creación de un programa ejecutable para su distribución e instalación en cuantos computadoras se desee, incluso aunque en ellos no esté instalado Visual Basic 6.0.



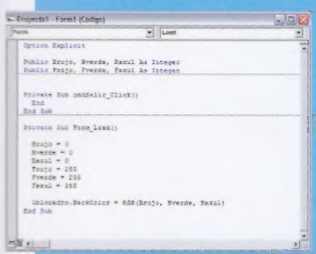
Para crear un programa ejecutable se utiliza el comando Generar nombreProyecto.exe en el menú Archivo. Así se generará un archivo con extensión *.exe. Para que este programa funcione en una computadora se necesita que el archivo MSVBVM60.DLL esté instalado en el directorio c:\Windows\System o c:\Winnt\System32. En proyectos más complejos que utilizan muchos controles pueden ser necesarios más archivos, la mayoría con extensiones *.ocx, *.vbx o *.dll. Para saber cuáles son los archivos necesarios se puede consultar el archivo *.vbp, que contiene la descripción completa del proyecto. Casi todos los archivos necesarios se instalan con el compilador de Visual Basic 6.0.



En el caso de que el programa se vaya a utilizar en una computadora sin Visual Basic, o de que en el proyecto se hayan utilizado controles comerciales (como los Crystal Reports para la creación de informes, los Sheridan Data Widgets o los True DBGrid de Apex para la gestión de bases de datos, etc.), es mejor construir disquetes de instalación que simplifiquen la tarea de instalar el programa. Visual Basic 6.0 dispone de un Asistente (Wizard) que simplifica la tarea de crear disquetes de instalación. Este Asistente está en el mismo grupo de programas que Visual Basic 6.0, y se llama Asistente para empaquetado y distribución.

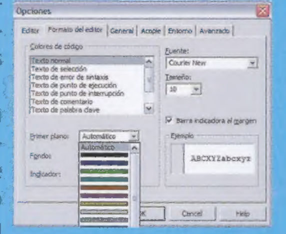
UTILIZANDO EL EDITOR DE CÓDIGO

El editor de código o Code Editor de Visual Basic 6.0 es la ventana en la cual se escriben las sentencias del programa. Esta ventana presenta algunas características muy interesantes que conviene conocer para sacar el máximo partido a la aplicación.



Para abrir la ventana del editor de código se elige Codigo en el menú Ver. También se abre al hacer clic en el botón Ver Código de la Ventana de Proyecto, o haciendo doble clic en el formulario, o en cualquiera de sus controles. Cada formulario, cada módulo de clase y cada módulo estándar tienen su propia ventana de código.

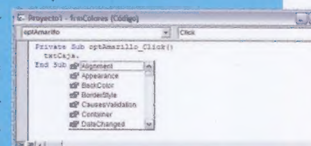
El Editor de Código de Visual Basic 6.0 ofrece muchas ayudas al usuario que requiere una explicación más detenida. En primer lugar, utiliza un código de colores (accesible y modificable en *Herramientas•Opciones•Formato de Editor*) para destacar cada elemento del programa. Así, el código escrito por el usuario aparece en negro, las palabras clave de Basic en azul, los comentarios en verde, los errores en rojo, etc.



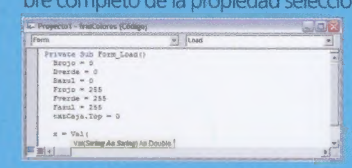
En la parte superior de esta ventana aparecen dos listas desplegables. La de la izquierda corresponde a los elementos del formulario (la parte General, común a todo el formulario; el propio formulario y los distintos controles incluidos en él). La lista desplegable de la derecha muestra los procedimientos que se corresponden con el elemento seleccionado en la lista de la izquierda. Estas dos listas permiten localizar fácilmente el código que se desee programar o modificar.

El código en la Figura 1.23 contiene una serie de declaraciones de variables y la opción de no permitir usar variables no declaradas (Option Explicit). Esta es la parte General de código del formulario. Aquí se puede definir funciones y procedimientos Sub no relacionados con un evento o control en particular. A continuación aparecen dos procedimientos Sub que se corresponden con los eventos Clic del botón cmdSalir y Load del formulario. Están separados por una línea, activada con Separador de Procedimientos en *Herramientas•Opciones•Editor.*

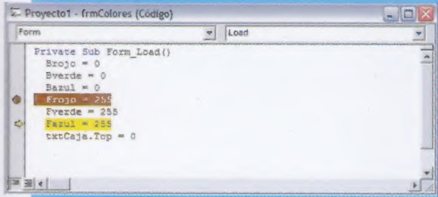
Para ver los procedimientos del formulario y sus controles simultáneamente en la misma ventana, o solo un procedimiento (seleccionado), se puede usar los pequeños botones en la parte inferior izquierda de la ventana. Son Ver Procedimiento y Ver módulo completo (accesibles en *Herramientas•Opciones•Editor*).



Otra opción interesante es completar automáticamente el código (Automatic Completion Code). Con esta opción, al teclear el punto (o la letra inicial de una propiedad después del punto) tras el nombre de un objeto, se abre una lista con las propiedades del objeto. Pulsando Tab se ingresa el nombre completo de la propiedad seleccionada. Esta característica es *AutoListMembers*. Además, la opción *AutoQuickInfo* hace que aparezca información sobre una función al teclear su nombre. AutoListMembers y AutoQuickInfo se activan en *Tools•Options•Editor.*



EJECUTANDO UN PROGRAMA DE MANERA CONTROLADA



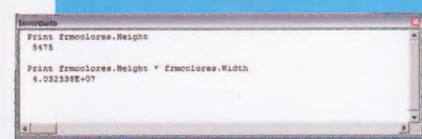
Para ejecutar parcialmente un programa se pueden utilizar varias formas. Una de ellas consiste en incluir Puntos de parada de ejecución (breakpoints) en determinadas líneas del código. Los breakpoints se indican con un punto grueso en el margen, y un cambio de color de la línea. En esta figura se muestra también la barra de herramientas Debug. El colocar un breakpoint en una línea de código implica que la ejecución del programa se detendrá al llegar a esa línea. Para insertar un breakpoint en una línea del código se utiliza la opción Alternar puntos de interrupción del menú Depuración, con el botón del mismo nombre o pulsando la tecla <F9>, estando el cursor posicionado sobre la línea en cuestión. Para borrarlo se repite esa operación. Cuando la ejecución está detenida en una línea aparece una flecha en el margen izquierdo, tal como puede verse también en la Figura 1.29. En ese momento se puede consultar el valor de cualquier variable que sea accesible desde ese punto en la ventana de depuración (Debug Window).

No es estrictamente necesaria la utilización de breakpoints para la ejecución parcial de un programa. Esto se puede hacer también ejecutando el programa paso a paso (o línea a línea). Para hacer esto hay varias opciones:

- **Pulsando la tecla <F8>:** Esta instrucción hace que se ejecute una línea del código. En el caso de que esta se trate de la llamada a un procedimiento o función, la ejecución se trasladará a la primera línea de ese procedimiento o función. En el caso de que se desee ejecutar toda la función en un único paso (por ejemplo, porque se tiene constancia de que esa función funciona correctamente), se puede hacer mediante la opción Paso a Paso por Procedimientos, pulsando las teclas <mayúsculas> y <F8> simultáneamente.
- **Ejecutar hasta el cursor o Ctrl+<F8>:** Esta instrucción ejecuta el programa hasta la línea en la que se encuentre posicionado el cursor.
- **Iniciar, botón o <F5>:** Permite continuar con la ejecución del programa hasta el siguiente breakpoint, o hasta el final del mismo.
- **Iniciar con compilación completa o Mayúsculas + <F5>:** Permite volver a comenzar la ejecución. Además de las mencionadas, existe la posibilidad de detener momentáneamente la ejecución del programa con el botón Interrumpir, o la combinación Ctrl+Pausa.

►►► DEPURANDO UN PROGRAMA

El Depurador de Visual Basic 6.0 dispone de varias formas para consultar el valor de variables y propiedades, así como para ejecutar funciones y procedimientos comprobando su correcto funcionamiento. En ello juegan un papel importante tres tipos de ventanas: Immediate, Locales y Watch.

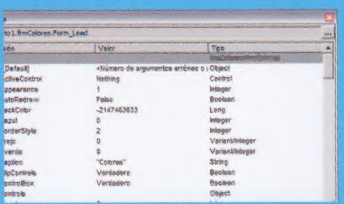


► **Ventana Immediate:** Permite realizar diversas acciones:

►► Imprimir el valor de cualquier variable y/o propiedad accesible la función o procedimiento que se está ejecutando: Esto se puede hacer utilizando el método Print VarName (o su equivalente ?VarName) directamente en dicha ventana, o introduciendo en el código del programa sentencias del tipo Debug.Print VarName. En este último caso, el valor de la variable o propiedad se escribe en la ventana Immediate sin necesidad de parar la ejecución del programa. Además, esas sentencias se guardan con el formulario, y no hay que volver a escribirlas para una nueva ejecución. Cuando se compila el programa para producir un ejecutable las sentencias Debug.Print son ignoradas.

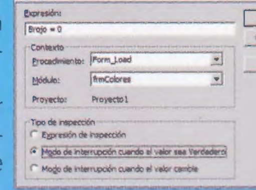
►► Asignar valores a variables y propiedades cuando la ejecución está detenida y proseguir la ejecución con los nuevos valores. Sin embargo, no se puede crear nuevas variables.

►► Ejecutar expresiones, y probar funciones y procedimientos, incluyendo en la ventana Immediate la llamada correspondiente.



► **Ventana Locales:** Muestra el valor de todas las variables visibles en el procedimiento en el que está detenida la ejecución.

► **Ventana Inspección Rápida:** Permite conocer permanentemente el valor de una variable sin tener que consultarlo cada vez.Para observar continuamente el valor de una variable o expresión, hay que añadirla a la ventana Inspecciones. Esto se hace con la opción Agregar Inspección del menú Depuración. El valor de las variables incluidas en la ventana Inspecciones se actualiza automáticamente, indicándose también cuando no son accesibles desde el procedimiento que se esté ejecutando (Out of Context).



La ventana Agregar Inspección permite introducir Breaks o paradas del programa condicionales, cuando se cumple cierta condición o cuando el valor de la variable cambia.

Las capacidades de Visual Basic 6.0 para vigilar el valor de las variables pueden activarse desde el menú Depuración o con algunos botones en la barra de herramientas Depuración.

► **Otras posibilidades del Depurador:** El Depurador de Visual Basic 6.0 permite no solo saber qué sentencia va a ser la próxima en ejecutarse (con Depuración.Mostrar siguiente instrucción), sino también decidir cuál va a ser dicha sentencia (con Depuración.Establecer siguiente instrucción), pudiendo cambiar de esta forma el curso habitual de la ejecución: saltando sentencias, volviendo a una sentencia ya ejecutada, etc.

►►► COMENTARIOS Y OTRAS UTILIDADES

► Visual Basic 6.0 interpreta que todo lo que está a la derecha del carácter (') en una línea cualquiera del programa es un comentario, y no lo tiene en cuenta para nada. El comentario puede empezar al comienzo de la línea o a continuación de una instrucción que debe ser ejecutada, por ejemplo:

```
' Esto es un comentario
A = B*x+3.4 ' también esto es un comentario
```

Los comentarios son tremendamente útiles para poder entender el código utilizado, facilitando de ese modo futuras revisiones y correcciones. En programas que no contengan muchas líneas de código puede no parecer demasiado importante, pero cuando se trata de proyectos realmente complejos, o desarrollados por varias personas su importancia es tremenda. En el caso de que el código no esté comentado, el trabajo de actualización y revisión puede resultar complicadísimo.

Otro aspecto práctico en la programación es la posibilidad de escribir una sentencia en más de una línea. En el caso de sentencias bastante largas es conveniente cortar la línea para que entre en la pantalla. En otro caso la lectura del código se hace mucho más pesada. Para ello es necesario dejar un espacio en blanco al final de la línea y escribir el carácter (_) tal y como se muestra en el siguiente ejemplo:

```
str1 = "Londres" : str2 = "París" 'Se inicializan las variables
Frase = "Me gustaría mucho viajar a " & _
str1 & " y a " & str2
'El contenido de Frase sería: "Me gustaría mucho viajar a Londres y a París
```

Una limitación a los comentarios en el código, es que no se pueden introducir en una línea en la que se ha introducido el carácter de continuación (_).

La sintaxis de Visual Basic 6.0 permite también incluir varias sentencias en una misma línea. Para ello las sentencias deben ir separadas por el carácter dos puntos (:). Por ejemplo:

```
m = a : n = b : resto = m Mod n ' Tres sentencias en una línea
```

►►► PROYECTOS Y MÓDULOS

Un proyecto realizado en Visual Basic 6.0 es el conjunto de todos los archivos o módulos necesarios para que un programa funcione. La información referente a esos archivos se almacena en uno del tipo ProjectName.vbp. La extensión *.vbp hace referencia a Visual Basic Project.

► Si se edita este archivo con cualquier editor de texto se comprueba que la información que almacena es la localización en los discos de los módulos que conforman ese proyecto, los controles utilizados (archivos con extensión .ocx), etc. En el caso más simple, un proyecto está formado por un único formulario y constará de dos archivos: el que define el proyecto (*.vbp), y el que define el formulario (*.frm).

► Los módulos que forman parte de un proyecto pueden ser de varios tipos: aquellos que están asociados a un formulario (*.frm), los que contienen únicamente líneas de código Basic (*.bas) llamados módulos estándar, y los que definen agrupaciones de código y datos denominadas clases (*.cls), llamados módulos de clase.

► Un módulo *.frm está constituido por un formulario y toda la información referente a los controles (y sus propiedades) en él contenidos, además de todo el código programado en los eventos de esos controles, y las funciones y procedimientos propios de ese formulario. En general, se llama función a una porción de código independiente que realiza una determinada actividad. En Visual Basic existen dos tipos de funciones: las llamadas function, que se caracterizan por tener valor de retorno, y los procedimientos o procedures, que no lo tienen. En otros lenguajes, como C/C++/Java, las function realizan los dos papeles.

► Un módulo de código estándar *.bas contendrá una o varias funciones y/o procedimientos, además de las variables que se desee, a los que se podrá acceder desde cualquiera de los módulos que forman el proyecto.

VARIABLES Y FUNCIONES

▸ **De ámbito local:** Un módulo puede contener variables y procedimientos, o funciones públicas y privadas. Las públicas son aquellas a las que se puede acceder libremente desde cualquier punto del proyecto. Para definir una variable, un procedimiento o una función como pública es necesario preceder a la definición de la palabra Public, como por ejemplo:

```
Public Variable1 As Integer
Public Sub Procedimiento1 (Parametro1 As Integer, ...)
Public Function Funcion1 (Parametro1 As Integer, ...) As Integer
```

▸ Para utilizar una variable Public, o llamar a una función Public, definidas en un formulario desde otro módulo, se debe preceder el nombre de la variable o procedimiento con el nombre del formulario al que pertenece, como por ejemplo:

```
Modulo1.Variable1
Call Modulo1.Procedimiento1(Parametro1, ...)
Retorno = Modulo1.Funcion1(Parametro1, ...)
```

▸ Sin embargo, si el módulo al que pertenecen la variable o el procedimiento Public es un módulo estándar (*.bas), no es necesario poner el nombre del módulo más que si hay coincidencia de nombres con los de otro módulo también estándar. Una variable Private, por el contrario, no es accesible desde ningún otro módulo distinto de aquél en el que se haya declarado.

▸ Se llama variable local a una variable definida dentro de un procedimiento o función. Las variables locales no son accesibles más que en el procedimiento o función en que están definidas.

▸ Una variable local es reinicializada (a cero, por defecto) cada vez que se entra en el procedimiento. Es decir, una variable local no conserva su valor entre una llamada al procedimiento y la siguiente. Para hacer que el valor de la variable se conserve hay que declarar la variable como static (como por ejemplo: Static n As Integer). Visual Basic inicializa una variable estática solamente la primera vez que se llama al procedimiento. Para declarar una variable estática, se utiliza la palabra Static en lugar de Dim. Un poco más adelante se verá que Dim es una palabra utilizada para crear variables. Si un procedimiento se declara Static, todas sus variables locales tienen carácter Static.

▸ **De ámbito global:** Se puede acceder a una variable o función global desde cualquier parte de la aplicación. Para hacer que una variable sea global, hay que declararla en la parte general de un módulo *.bas, o de un formulario de la aplicación. Para declarar una variable global se utiliza la palabra Public. Por ejemplo:

```
Public var1_global As Double, var2_global As String
```

▸ De esta forma se podrá acceder a las variables var1_global, var2_global desde todos los formularios. La tabla muestra la accesibilidad de las variables en función de dónde y cómo se hayan declarado.
La diferencia entre las variables y/o procedimientos Public de los formularios y de los módulos estándar, está en que las de los procedimientos deben ser cualificadas (precedidas) por el nombre del formulario cuando se llaman desde otro módulo distinto, mientras que las de un módulo estándar (*.bas) solo necesitan ser cualificadas si hay colisión o coincidencia de nombres.

▸ **Nota:** Las palabras Global y Dim proceden de versiones antiguas de Visual Basic, y debe preferirse la utilización de las palabras clave Public y Private, que expresan mejor su significado.

Tipo de variable	Lugar de declaración	Accesibilidad
Global o Public	Declaraciones de *.bas	Desde todos los formularios
Dim o Private	Declaraciones de *.bas	Desde todas las funciones de ese módulo
Public	Declaraciones de *.frm	Desde cualquier procedimiento del propio formulario, y desde otros precedida del nombre del módulo en el que se ha declarado
Dim o Private	Declaraciones de *.frm	Desde cualquier procedimiento del propio formulario
Dim	Cualquier procedimiento de un módulo	Desde el propio procedimiento

PROYECTOS Y MÓDULOS

Menú Archivo

TECLA	ACCIÓN
Ctrl+N	Nuevo Proyecto
Ctrl+O	Abrir Proyecto
Ctrl+S	Guardar Formulario
Ctrl+P	Imprimir
Alt+Q	Salir

Menú Ver

TECLA	ACCIÓN
Mayús+F7	Objeto
Mayús+F2	Definición
Ctrl+Mayús+F2	Ultima posición
F2	Examinador de Objetos
Ctrl+G	Ventana Inmediato
Ctrl+R	Explorador de Proyectos
F4	Ventana de Propiedades
Mayús+F4	Pagina de propiedades

Menú Edición

TECLA	ACCIÓN
Ctrl+Z	Deshacer
Ctrl+X	Cortar
Ctrl+C	Copiar
Ctrl+V	Pegar
Del	Eliminar
Ctrl+A	Seleccionar todo
Ctrl+F	Buscar
F3	Buscar Siguiente
Ctrl+H	Reemplazar
Ctrl+J	Mostrar propiedades y métodos
Ctrl+Mayús+J	Mostrar Constantes
Ctrl+I	Información rápido
Ctrl+Mayús+I	Información de parámetros
Ctrl+Alt+A	Palabra Completa

Menú Proyecto

TECLA	ACCIÓN
Ctrl+D	Agregar Archivo
Ctrl+T	Componentes

Menú Depuración

TECLA	ACCIÓN
F8	Paso a paso por instrucciones
Mayús+F8	Paso a paso por procedimientos
Ctrl+F8	Ejecutar hasta el cursor
Mayús+F9	Inspección rápida
F9	Alternar puntos de interrupción
Ctrl+Mayús+F9	Borrar los puntos de interrupción

Menú Ejecutar

TECLA	ACCIÓN
F5	Iniciar
Ctrl+F5	Iniciar con compilación rápida

▸ DIRECTOR: BERNARDO ROCA REY MIRÓ QUESADA ▸ EDITORA: FRANCA CAVASSA ▸ EDITORA ADJUNTA: MELISA OSORES ▸ DESARROLLO: LUIS FIGALA / HERNÁN BALDERRAMA ▸ GERENTE COMERCIAL: AUGUSTO TOWSEND ▸ GERENTE DE PRODUCTO: RENZO MARIÁTEGUI ▸ JEFE DE PRODUCTO: PAOLA CORNEJO ▸ COORDINACIÓN: DORA NIQUEN ▸ EDITORA DE DISEÑO: CLAUDIA BURGA-CISNEROS ▸ DISEÑO: VERUZKA NORIEGA ▸ DIAGRAMACIÓN: GERARDO CRISTÓBAL ▸ EDICIÓN, PREPRENSA E IMPRESIÓN: EMPRESA EDITORA EL COMERCIO S.A., PERÚ, AÑO 2004